UNIVERSIDADE FEDERAL DO ABC

ALLAN HOLANDA TARGINO

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UM PROGRAMA DE TRANSCRIÇÃO DE PORTUGUÊS BRASILEIRO PARA BRAILLE VISUAL

Santo André

2016

UNIVERSIDADE FEDERAL DO ABC

ALLAN HOLANDA TARGINO

DESENVOLVIMENTO E IMPLEMENTAÇÃO DE UM PROGRAMA DE TRANSCRIÇÃO DE PORTUGUÊS BRASILEIRO PARA BRAILLE VISUAL

Proposto de projeto apresentado à disciplina de Trabalho de Graduação da Universidade Federal do ABC.

Orientador: Ricardo Suyama.

Santo André

2016

Lista de Figuras

Figura 1 – Cela Braille..... 7
Figura 2 – Funcionamento macro do motor de reconhecimento e conversão de Inks em caracteres..... 11
Figura 3 – Fluxograma de reconhecimento e conversão de texto proveniente de um Ink da biblioteca utilizada..... 12
Figura 4 – Reconhecimento de palavras associado a identificação de contextos mais prováveis de existência..... 13
Figura 5 – Estrutura a ser renderizada em XAML para posterior preenchimento dos bits de um caracter..... 14
Figura 6 – Diagrama hierárquico da Cela Braille..... 16
Figura 7 -Diagrama de funcionamento da conversão de letras para uma coleção de pontos Braille..... 17
Figura 8 – Classe MainPage..... 17
Figura 9 – Classe BraillePoint..... 18
Figura 10 – Classe BrailleCell..... 18
Figura 11 – Classe BrailleUtil..... 19
Figura 12 – Página principal da aplicação..... 20
Figura 13 – Exemplo 01 de reconhecimento de texto em linguagem natural e tradução em Braille..... 21
Figura 14 - Exemplo 02 de reconhecimento de texto em linguagem natural e tradução em Braille..... 21
Figura 15 – Composição da letra A..... 22
Figura 16 – Método para determinar a intersecção de strokes..... 23
Figura 17 – Palavras “Allan Holanda” usadas no entendimento do modelo matemático a ser aplicado..... 23
Figura 18 – Gráfico da distância entre palavras sem qualquer tratamento..... 24
Figura 19 - Gráfico da distância entre palavras após a filtragem de mediana..... 26
Figura 20 – Inserção de celas Braille logo após o início de cada palavra distinta..... 26

Figura 21 - Inserção de celas Braille logo após o início de cada palavra distinta vista no contexto da tela do programa. 27
Figura 22 – Tela de geração de documentos e impressão de uma sessão de tradução Braille. 28

Sumário

1. Introdução 6

1.1. Definições e Contextualização 7

2. Objetivos 9

3. Proposta de Execução 10

3.1. Reconhecimento de escrita em Português 10

3.2. Tradutor de Português para Braille Visual 13

3.3. Gerador de documentos em Braille prontos para impressão 14

3.4. Suporte a extensibilidade e futuras expansões 15

4. Resultados 16

4.1. Modelagem de classes 16

4.2. Desenho Inicial da Aplicação 19

4.3. Desenho final da Aplicação 22

5. Cronograma de Execução 29

6. Referências 30

7. Anexos 32

7.1. Códigos-Fonte 32

7.1.1. MainPage.xaml 32

7.1.2. MainPage.xaml.cs 33

7.1.3. BraillePoint.cs 39

7.1.4. BrailleCell.xaml 39

7.1.5. BrailleCell.xaml.cs 40

7.1.6. BrailleUtil.cs 41

7.1.7. LetterBrailleConverter.cs 41

1. INTRODUÇÃO

No Brasil, existem mais de 6,5 milhões de pessoas com algum tipo de deficiência visual, sendo 582 mil cegas e seis milhões com baixa visão, segundo dados da fundação com base no Censo 2010, feito pelo Instituto Brasileiro de Geografia e Estatística (IBGE 2010).

Dentro de escolas, alunos com algum tipo de deficiência visual encaram diariamente problemas no consumo e absorção do conteúdo ensinado, possuindo desde problemas graves de aprendizado como também de inserção e relacionamento com outras crianças. Estudantes que possuem deficiência visual devem ser amparados por um corpo docente e educacional que possa suportar as mais diversas necessidades que eles possam ter, o que exige treinamento e técnicas efetivas de ensino por parte deste time de profissionais. (The American Foundation for the Blind's Josephine L. Taylor Leadership Institute 2000)

Além de uma equipe bem preparada é necessário que se tenha ferramentas adequadas que sejam inclusivas o suficiente para que o aluno possa aprender com efetividade o conteúdo desejado. Hoje são utilizados desde livros e materiais em uma forma de mídia inclusiva (como por exemplo, o Braille ou áudio) quando serviços de inclusão especializados. Através desta combinação de profissionais e ferramentas adequadas, pode-se permitir que o aluno com deficiências visuais possa competir efetivamente ao máximo com seus colegas de classe que não possuem qualquer deficiência. (The American Foundation for the Blind's Josephine L. Taylor Leadership Institute 2000)

Porém, isso é particularmente problemático dado que ter uma equipe de apoio capacitada e ter as ferramentas necessárias de inclusão disponíveis da escola, dependendo da realidade econômica em que a instituição se encontra, pode elevar os custos da implementação a níveis fora do que a mesma pode arcar.

Neste cenário, uma alternativa que contemple custos acessíveis e dispense parcialmente uma grande equipe de apoio ao deficiente visual faz-se necessário. Através do presente trabalho propõe-se o desenvolvimento de um aplicativo a ser usados por alunos e professores a fim de superar e facilitar o consumo de conteúdo em português através do Braille visual, montando assim uma arquitetura onde o aluno possa estudar durante e após o conteúdo da aula ser dado.

1.1. DEFINIÇÕES E CONTEXTUALIZAÇÃO

Define-se Braille como um sistema de escrita e leitura tátil para pessoas cegas ou com alguma deficiência visual grave. Surgiu em 1825 na França, sendo seu criador Louis Braille, um homem que ficou cego aos três anos de idade vítima de um acidente. (Apadev s.d.)

As letras em Braille são formadas a partir da combinação de 6 pontos que compõem o que é chamado de Cela Braille. Um exemplo de Cela Braille pode ser visto na figura 1, disposta a seguir. A cela é formada por 3 linhas e 2 colunas de pontos. A numeração dos pontos começa a partir da primeira coluna, à esquerda, numerando-os de cima para baixo, seguido da numeração da segunda coluna, à direita, também de cima para baixo. (Nicolaiewsky and Correa 2008)

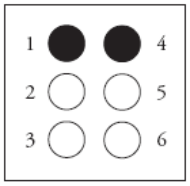

Figura 1 – Cela Braille

Por serem formadas a partir da combinação de seis pontos, as letras em Braille são bem mais semelhantes entre si do que as letras do sistema impresso regular. Logo o aprendizado do sistema Braille é muito mais difícil de ser aprendido do que o comum. (Pring 1994)

O Braille é um dos principais meios de acesso à informação impressa pelos quais estudantes com algum tipo de deficiência visual em grau avançado podem ter. Regina Oliveira, coordenadora da Fundação Dorina Nowill, uma entidade responsável por difundir o Braille no Brasil, afirma que “Com o Braille as pessoas cegas passaram a ter acesso ao conhecimento, à cultura, ao lazer, à informação [...]”. Além disso, para certas aplicações, o Braille se faz extremamente necessário ou não há qualquer outra alternativa a ele, como afirma Regina ao dizer que “O Braille é imprescindível para alfabetização das crianças, para que elas tenham contato com a ortografia, tanto da língua portuguesa quanto de línguas estrangeiras. Para livros científicos, não existe um substituto pro Braille ainda.”. (Portal Brasil 2015)

Pesquisadores afirmam que estudantes que estão aprendendo Braille possuem necessidades, habilidades e interesses diversos, logo professores que usam Braille devem usar uma variedade de estratégias educacionais com seus estudantes. (Holbrook and Nannen 1997)

Define-se Braille Visual como a exposição de um conjunto de letras Braille seja em sua forma impressa e não tátil ou através de meios digitais. Da mesma maneira como o Braille transmite de forma tátil as informações através de seu alfabeto, o Braille visual atinge o mesmo objetivo usando este mesmo alfabeto de maneira gráfica. Dado que em escolas há a tanto a presença de crianças com ou sem deficiências visuais, faz-se necessário adotar metodologias eficientes de ensino que contemple ambas necessidades de aprendizado. Sabe-se que o Braille Visual pode integrar de maneira mais fluida e natural a interação de ensino entre tais alunos.

Particularmente, para uma criança cega ou com alguma deficiência visual grave em fase escolar, há uma série de desafios de aprendizado a serem superados, em que estas acabam desempenhando de maneira pior do que colegas com visão. Há muitos conceitos que não são completamente compreensíveis sem a visão, como por exemplo, o que significa ser alto? Como explicar a diferença entre um homem alto, uma árvore alta e um prédio alto, para alguém que nunca viu? Descrever nuvens brancas em um céu azul? (Rao s.d.) Professores que trabalham com crianças cegas devem ajudá-las a compreender e implementar muitos conceitos e habilidades básicas, num meio onde convivem alunos com ou sem deficiência, onde o Braille Visual pode ser um meio interessante de integração para todos esses alunos aprendem de maneira efetiva.

Para o presente trabalho, propõe-se uma maneira alternativa de interação entre alunos e professores através de um software de transcrição de português em Braille a fim de melhorar a problemática citada. O software será capaz de promover uma boa experiência em sala de aula onde a escrita do professor seja reconhecida em tempo real e seja transcrita em Braille Visual. Ao final da aula, é gerado um arquivo com a transcrição da aula a ser enviado para impressoras Braille, de modo que os alunos postam estudar e revisar o conteúdo posteriormente. Além disso, também se espera que assim que novos dispositivos eletrônicos de geração dinâmica e tátil de Braille sejam produzidos, que estes se integrem facilmente com a solução proposta.

2. OBJETIVOS

Objetivo Geral:

- Tornar o aprendizado mais eficiente para os alunos portadores de deficiência em classes de aula, permitindo uma maior integração entre todos alunos envolvidos. Além disso, desenvolver uma solução de baixo custo e que seja viável do ponto de vista de implementação.

Objetivos Específicos:

- Usar um mecanismo de identificação de caracteres manuscritos em português e transcrever o resultado para Braille visual, dispondo-o e exibindo-o em um aplicativo Windows;
- Gerar uma síntese de uma aula de aula transcrita em Braille Visual pronta para a respectiva impressão em Braille por equipamento específico;
- Desenvolver suporte a extensão do aplicativo em questão para que possíveis integrações de hardware possam ser adicionadas.

3. PROPOSTA DE EXECUÇÃO

Dado a motivação e fundamentação teórica apresentadas anteriormente, propõe-se o desenvolvimento de um software que ajude na ambientação e melhor relacionamento durante as aulas entre professores, alunos sem deficiências visuais e alunos portadores de deficiências visuais em algum grau. Através do uso deste software em sala de aula, espera-se que o processo de aprendizado seja mais eficiente para os alunos portadores de deficiência, bem como que possa haver uma maior integração entre todos alunos.

O software consistirá em 4 módulos principais:

- Reconhecimento de escrita em Português;
- Tradutor de Português para Braille Visual;
- Gerador de documentos em Braille prontos para impressão;
- Suporte a extensibilidade e futuras expansões.

A seguir são descritos detalhadamente cada um dos módulos acima.

3.1. RECONHECIMENTO DE ESCRITA EM PORTUGUÊS

OCR (Acrônimo de Optical Character Recognition, ou Reconhecimento Óptico de Caracteres) refere-se a ambas tecnologia e processo aplicados na conversão de escrita natural ou impressa em texto codificado em linguagem de máquina, de maneira que um computador ou sistema computacional possa processá-lo. É um subconjunto das técnicas de processamento digital e reconhecimento de imagens, sendo amplamente usado em documentos de bancos, notas fiscais, passaportes e outros documentos de uso comercial. (Techopedia n.d.) Neste módulo do projeto serão usados técnicas de OCR e Aprendizado de máquina para transcrever a escrita natural do professor em uma cadeia de caracteres digitais.

Foi escolhido se desenvolver um aplicativo sobre a plataforma do *Universal Windows Platform* (UWP) (Whitney 2016), usando a linguagem de programação C# e a linguagem de descrição de interface XAML. Um dos principais motivos da escolha desta plataforma é o fato dos aplicativos rodarem no sistema operacional Windows. Além disso, pelo fato de serem

aplicativos universais, o mesmo aplicativo pode rodar em TVs, Tablets, Celulares e outros devices que rodem o sistema operacional Windows. Além disso, há bibliotecas de reconhecimento de escrita com altas taxas de acerto presentes no ambiente de desenvolvimento escolhido.

Para a realização da transcrição, será usada a classe *InkCanvas* (Microsoft, InkCanvas class 2016), usada em conjunto com o XAML, e o *namespace Windows.UI.Input.Inking* (Microsoft, Windows.UI.Input.Inking namespace 2016), usado basicamente em C#. Há o método *RecognizeAsync* presente em uma das classes, responsável pelo reconhecimento propriamente dito. O resultado da transcrição de escrita natural em tempo real é passado ao módulo Tradutor de Português para Braille Visual, descrito a seguir.

A descrição do algoritmo de reconhecimento e sua respectiva implementação são restritos conforme descrita na solicitação da patente de nome "CONVERTING DIGITAL INK TO SHAPES AND TEXT", do ano de 2009. (Microsoft Corporation 2009) Tal patente apenas contém dados do funcionamento macro do algoritmo e do sistema reconhecedor de *Inks*, conforme descrito na figura 2.

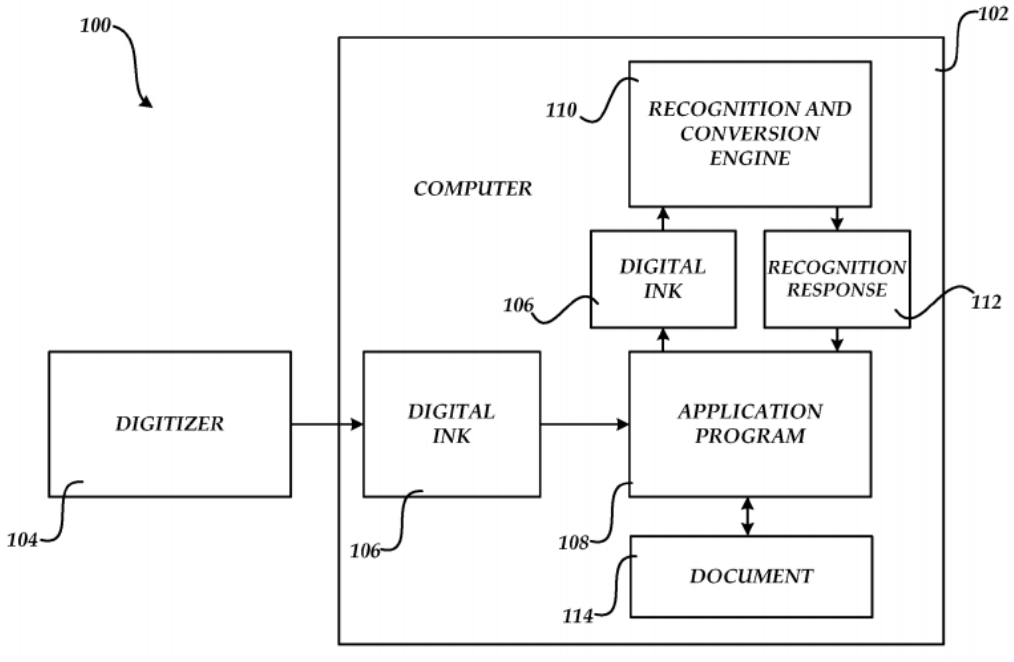

Figura 2 – Funcionamento macro do motor de reconhecimento e conversão de Inks em caracteres.

O fluxograma que engloba desde a entrada de dados até a inserção de textos por um programa cliente que consume o motor de reconhecimento e conversão de textos, originalmente descrito na patente e reproduzido aqui, é mostrado logo abaixo.

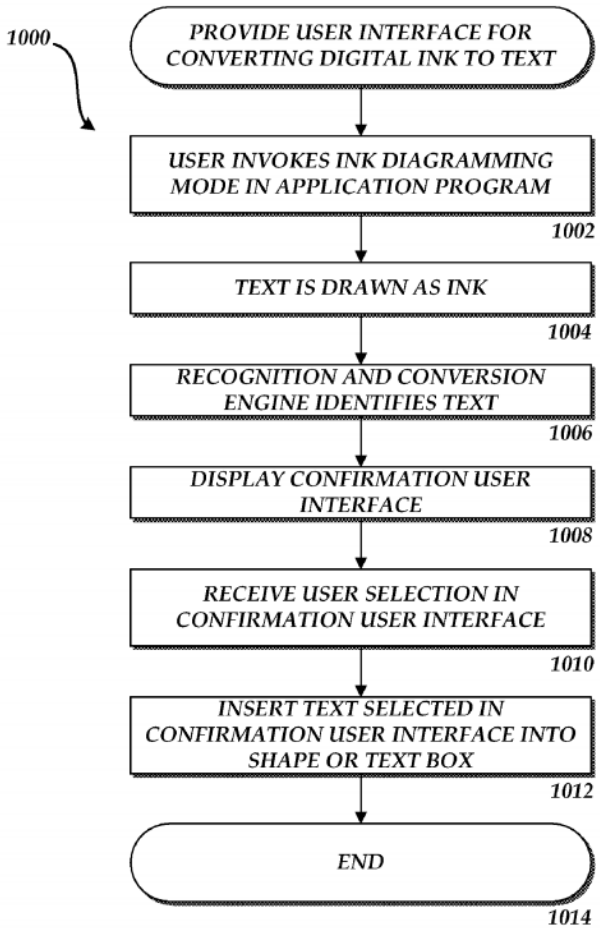

Figura 3 – Fluxograma de reconhecimento e conversão de texto proveniente de um Ink da biblioteca utilizada.

Uma característica bem interessante e útil da implementação específica desta biblioteca inclui técnicas associativas de Aprendizado de Máquinas para, não somente transcrever unicamente e identificar caracteres, mas também identificar contextos aos quais a composição dos caracteres ou palavras está inserido, resultando em um reconhecimento com melhores resultados. Esse comportamento é exemplificado na figura 4.

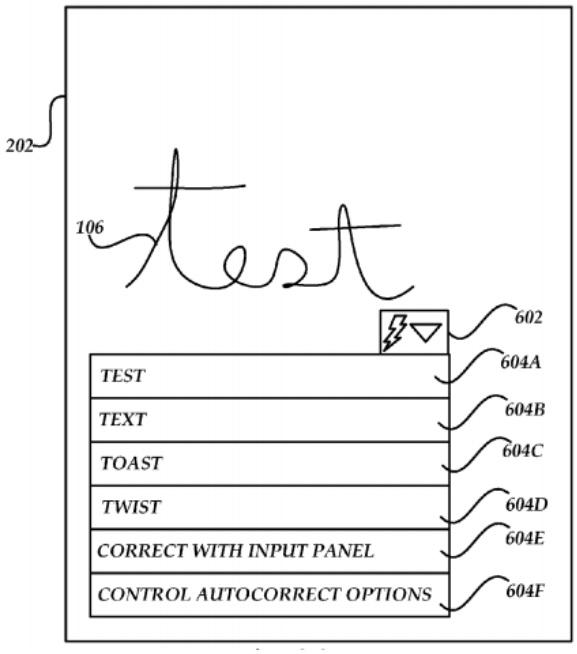

Figura 4 – Reconhecimento de palavras associado a identificação de contextos mais prováveis de existência.

Segundo a documentação oficial, a biblioteca suporte o reconhecimento de 30 idiomas. Baseado na acurácia e performance, as línguas suportadas são divididas em 3 categorias: Excelente, Muito Boa ou Boa. O Português enquadra-se na categoria Excelente, favorecendo o cenário do presente trabalho. (Microsoft n.d.).

O motor de OCR possui algumas limitações e especificações definidas em sua documentação. Ele espera e assume que o texto esteja escrito (em forma de *Ink*) em uma mesma direção – caso não esteja, parte do texto pode não ser reconhecido corretamente. A qualidade da imagem e um respectivo ruído que a mesma tenha podem distorcer os resultados esperados, dado que imagens não-nítidas, uso de fontes artísticas, fontes muito pequenas (menores que 15px), presença de planos de fundos complexos, sombras ao redor do texto, caracteres subscritos ou caracteres sobescritos podem prejudicar o desempenho do motor de OCR.

3.2. TRADUTOR DE PORTUGUÊS PARA BRAILLE VISUAL

Uma vez recebido o resultado do processo anterior, o módulo de tradução para Braille Visual fará a busca em uma tabela interna ao software para determinar quais conjuntos de cadeias

de bits equivalem ao texto em questão. Uma vez feito esta correlação, o software irá utilizar templates XAML para desenhar os caracteres em Braille. O template XAML será responsável renderizar a estrutura da figura 1.

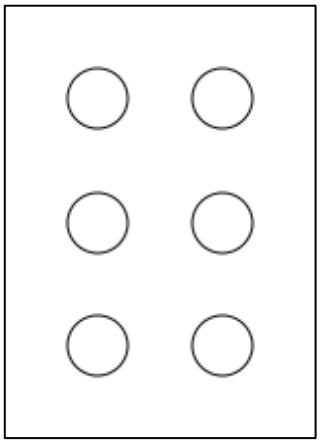

Figura 5 – Estrutura a ser renderizada em XAML para posterior preenchimento dos bits de um caracter.

Foi escolhido renderizar dinamicamente os caracteres a fim de se obter uma melhor performance de vídeo e armazenamento, ao invés de se armazenar todos caracteres possíveis em forma de imagem e em seguida mostra-las na tela. Além disso, pode-se redimensionar e processar tais caracteres de maneira nativa, sem precisar de bibliotecas adicionais de processamento digital de sinais.

3.3. GERADOR DE DOCUMENTOS EM BRAILLE PRONTOS PARA IMPRESSÃO

Uma vez que uma aula acabe e se tenha todo um mapeamento de português para os respectivos caracteres em Braille, o módulo em questão será responsável por gerar um documento PDF que esteja pronto para ser impresso impressoras Braille. Apenas as anotações que contenham textos serão convertidas, não sendo incluídos desenhos e imagens, por exemplo. Isso possibilita que alunos portadores de deficiência possam estudar as aulas após as mesmas terem acabado.

3.4. SUPORTE A EXTENSIBILIDADE E FUTURAS EXPANSÕES

Por fim, propõe-se que o software seja facilmente utilizado por futuras expansões. Isso será implementado através de comunicação serial, o qual a cada transcrição de escrita natural, caso haja alguém conectado através de uma porta serial, a cadeia de bits relativa aos caracteres em Braille será enviada. Um protocolo muito simples de comunicação será desenvolvido, apenas para garantir a entrega e a integridade da mensagem ao consumidor.

Isto é particularmente interessante pois uma vez que haja dispositivos eletrônicos que possam fisicamente expressar o Braille de maneira dinâmica e em tempo real, o software em questão enviará a cadeia de bits e o aluno portador de deficiência visual poderá ter interações muito mais rápidas dentro de sala de aula, possibilitando assim um aprendizado muito mais efetivo.

4. RESULTADOS

4.1. MODELAGEM DE CLASSES

Optou-se por desenvolver o software do presente Projeto de Graduação utilizando-se do paradigma computacional de Orientação à Objetos. Logo, foi necessário modelar o sistema quanto as classes e funcionalidades a serem utilizadas. O resultado da lógica dessa modelagem inicial é disposto nas figuras 6 e 7 a seguir.

Uma *BrailleCell* seria composta de uma coleção de *BraillePoint*. Optou-se por usar uma lista ligada no encadeamento destes pontos ao invés de um vetor de índice direto por questões de melhor integração com a parte gráfica da aplicação (telas desenhadas em XAML). Cada lista ligada possui um tamanho fixo de 6 elementos, os quais eram pré-alocados novamente pensando em performance, uma vez que a escrita de celas Braille em tempo real na tela poderiam consumir bastante processamento, culminando em uma má experiência de reconhecimento e tradução próximo de tempo real.

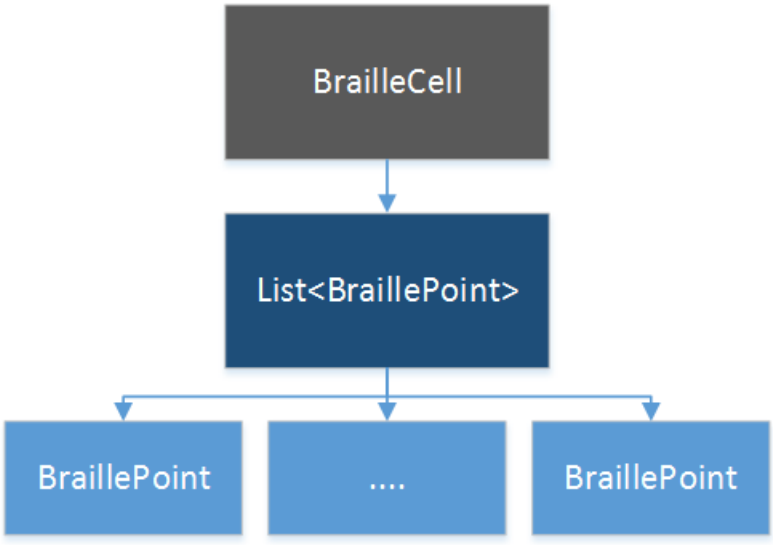

Figura 6 – Diagrama hierárquico da Cela Braille

Em seguida, foi necessário modelar a lógica envolvendo a conversão e mapeamento de caracteres para um conjunto de 6 pontos Braille. Essa responsabilidade foi atribuída a classe *BrailleUtil*, o qual consulta o mapeamento presente em *LetterBrailleConverter* e retorna os respectivos pontos associados a um caractere.

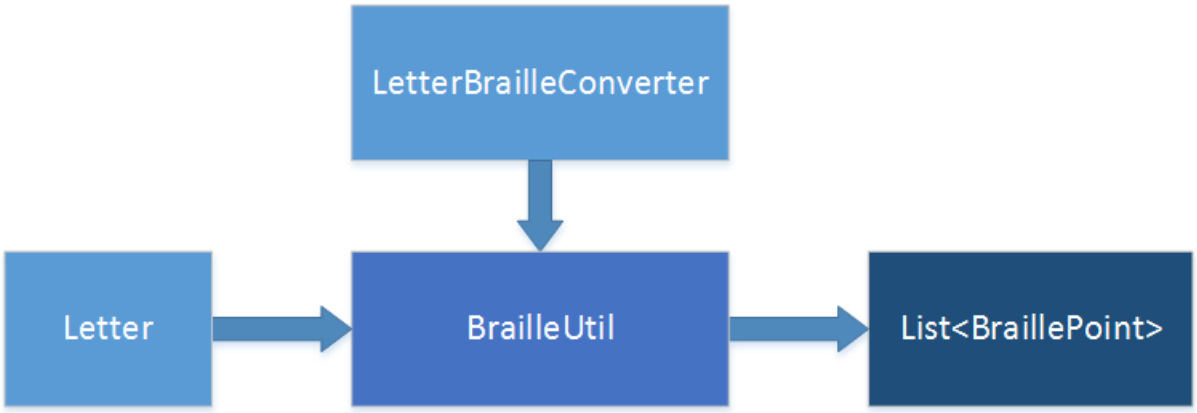

Figura 7 -Diagrama de funcionamento da conversão de letras para uma coleção de pontos Braille.

Com esta lógica desenhada e projetada, começou-se a implementação das classes acima, cujo diagrama de classes contendo a assinatura dos métodos e propriedades estão dispostas nas figuras 8, 9, 10 e 11. Foram omitidas as classes *LetterBrailleConverter* e *App*, uma vez que a primeira apenas continha campos de mapeamento e a segunda era a classe padrão de desenvolvimento na plataforma de aplicativos universais do Windows (UWP).

A classe *MainPage* é responsável por ser o grande container que abriga tanto o design da aplicação, o reconhecimento do texto e a manipulação dinâmica de celas Braille. O código-fonte do design pode ser encontrado no anexo 7.1 ao passo que o respectivo *code-behind* no anexo 7.2

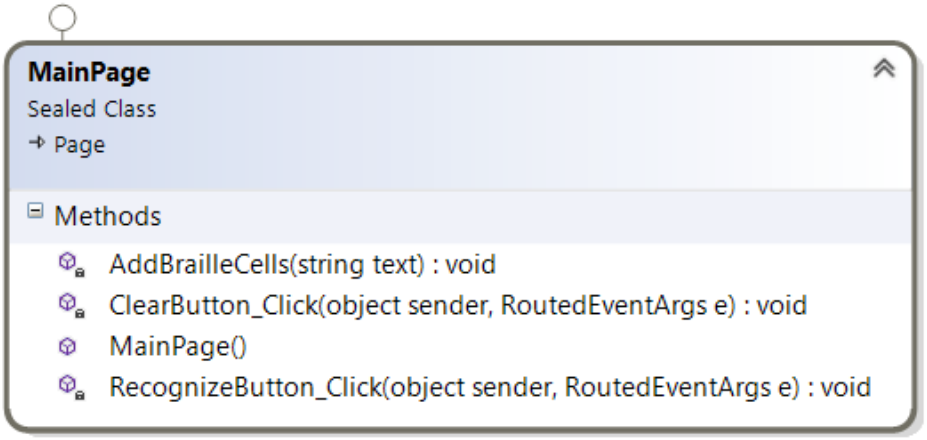

Figura 8 – Classe MainPage

A classe BraillePoint apenas contém uma propriedade relativa a cor que será exibida seu ponto. Por padrão, atribui-se a cor preta (#FFFFFF) a um ponto que seja preenchido e a cor branca a um ponto (#000000) que não seja preenchido. O código fonte pode ser encontrado no anexo 7.1.3.

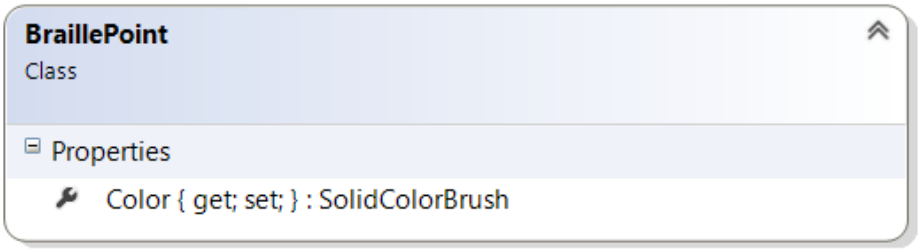

Figura 9 – Classe BraillePoint

A classe *BrailleCell*, como mostrado na lógica anterior, contém um conjunto de *BraillePoint*, os quais são obtidos dentro do construtor da mesma. O respectivo código fonte do design pode ser encontrado no anexo 7.1.4, enquanto o seu *code-behind* no anexo 7.1.5.

Figura 10 – Classe BrailleCell

Por fim, a classe BrailleUtil também implementa o desenvolvimento da lógica expandida anteriormente. O código-fonte pode ser encontrado no anexo 7.1.6.

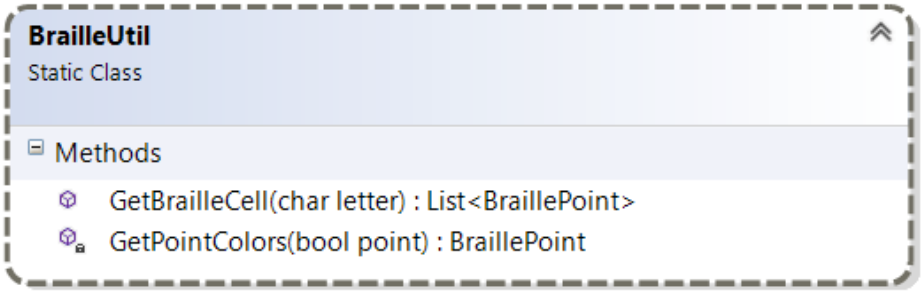

Figura 11 – Classe BrailleUtil

4.2. DESENHO INICIAL DA APLICAÇÃO

A página principal da aplicação é mostrada na figura 12. Foram criadas 4 regiões principais neste layout. A primeira remete ao painel de escrita natural em português simulando a escrita de um professor em sala de aula. A segunda região mostra o resultado do reconhecimento da escrita em português, para que o escritor possa conferir a veracidade e confiabilidade da tradução em tempo real. A terceira região contém um pequeno painel de botões usados para operar o aplicativo, contendo os botões “Reconhecer”, “Apagar”, “Serial” e “PDF”. A última região contém a coleção de celas Braille relativa a um reconhecimento prévio do texto escrito em linguagem natural.

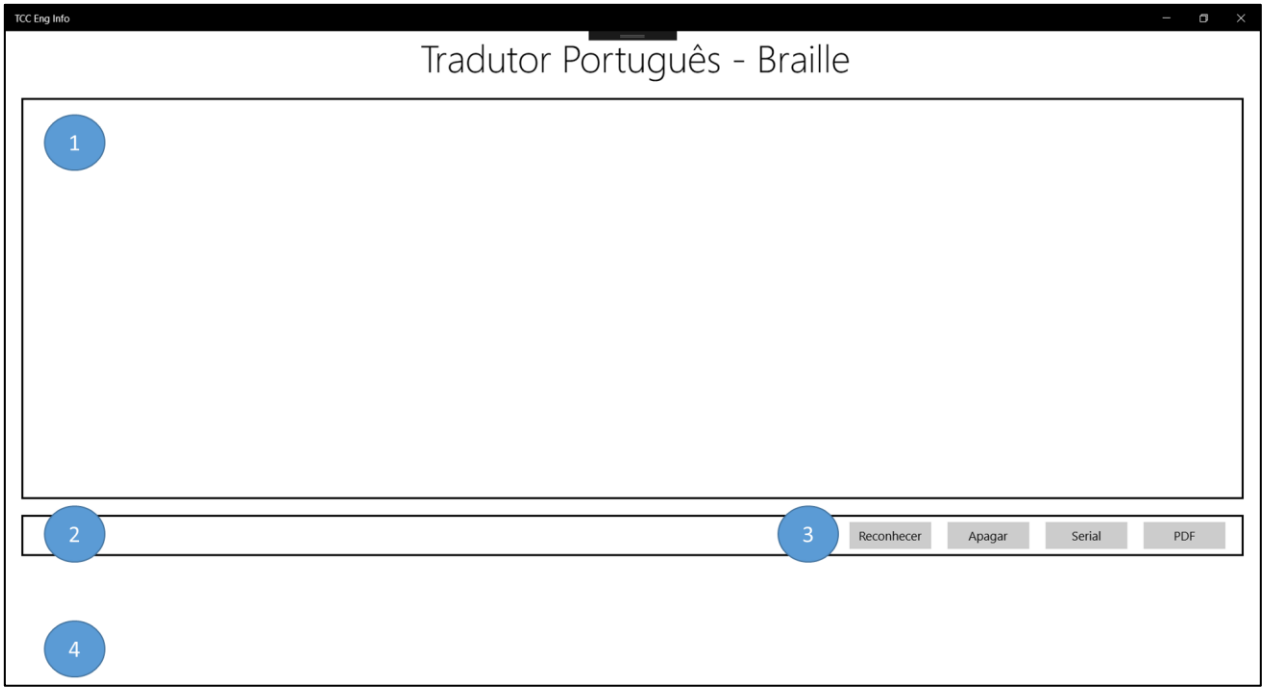

Figura 12 – Página principal da aplicação

Alguns exemplos de reconhecimento de escrita natural e respectivas traduções são apresentadas nas figuras 13 e 14, dispostas a seguir. Pode-se notar que a maneira como se escreve pode ser realizada através de letras denominadas informalmente como estilo “Forma”, “Bastão” ou mesmo “de Mão”.

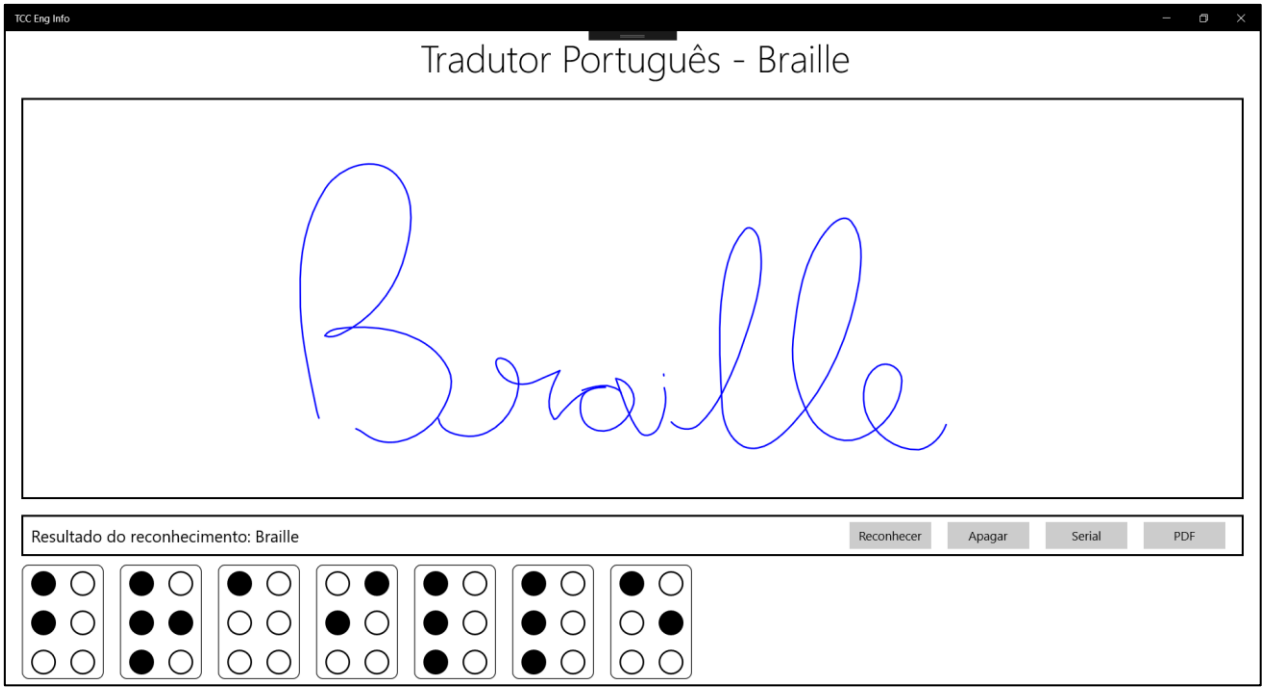

Figura 13 – Exemplo 01 de reconhecimento de texto em linguagem natural e tradução em Braille

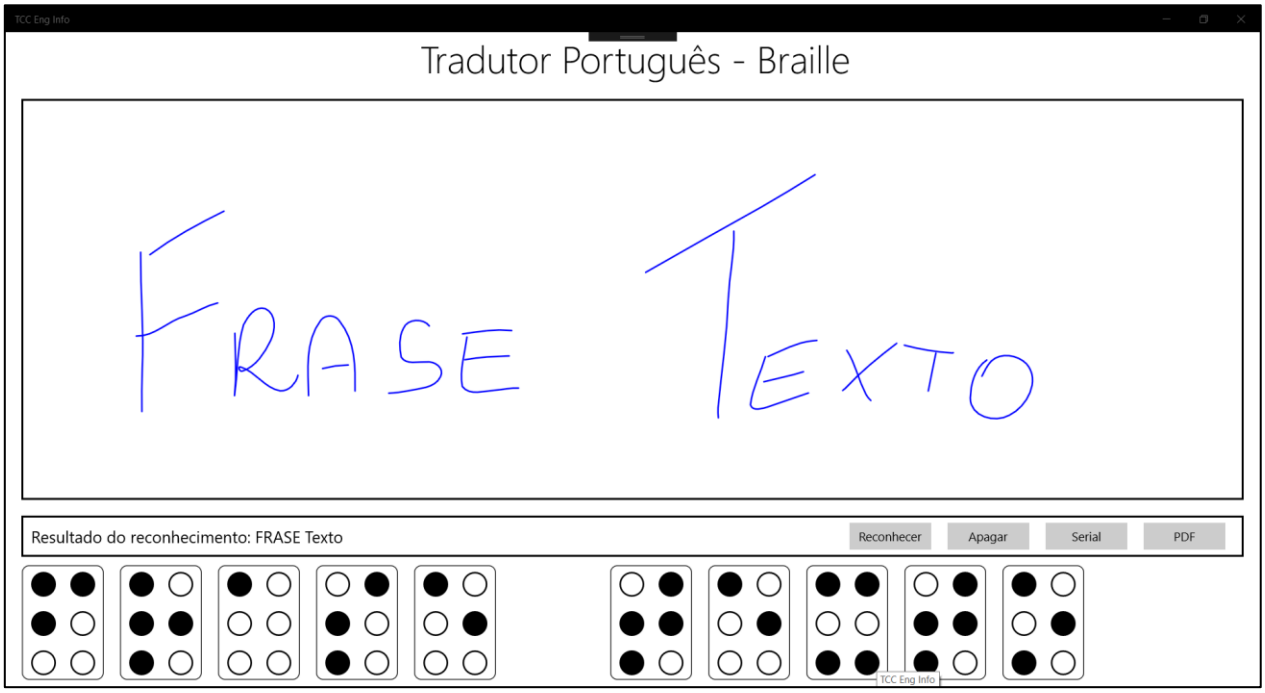

Figura 14 - Exemplo 02 de reconhecimento de texto em linguagem natural e tradução em Braille

4.3. Desenho final da Aplicação

Um importante feedback dado pelo orientador do presente Trabalhado de Graduação foi que seria interessante que se adicionasse as células Braille logo abaixo do texto escrito, possibilitando que os alunos com baixa visão tivessem a tradução de português da maneira como o professor escrevesse na lousa digital. Para isso, foi necessário desenvolver um modelo matemático simples que compreendesse esse fim.

Uma característica do componente utilizado é que a cada traço que o usuário faça, denominado internamente de *stroke*, ele guarda este traço como um elemento individual. O problema ocorre em letras que tipicamente necessitam de dois ou mais traços, como por exemplo a letra "A", conforme mostra a figura 15 abaixo.

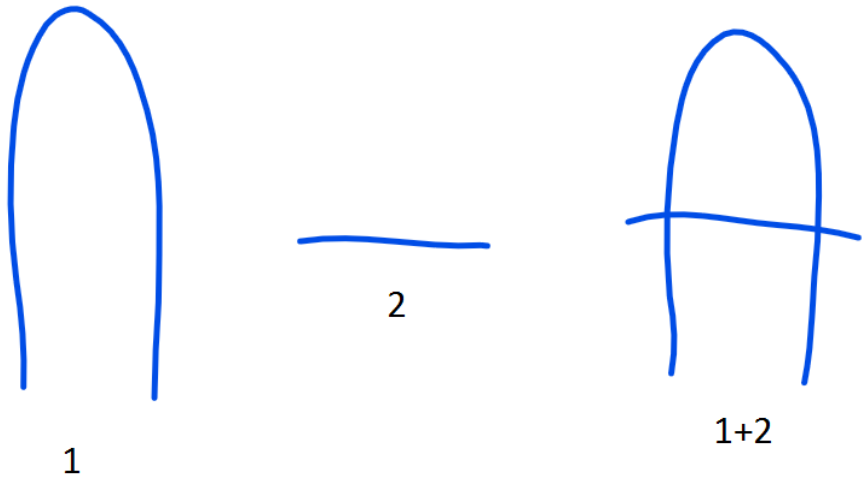

Figura 15 – Composição da letra A

O primeiro processamento (antes de se preocupar puramente com o modelo matemático para posicionar as células Braille) deveria envolver detectar quando uma letra seria formada. Como o SDK utilizado devolvia um retângulo em torno de um *stroke*, conforme a figura 16, optou-se por verificar quando acontecia uma intersecção da área dos retângulos. Quando a intersecção era vazia entre strokes adjacentes, tratavam-se de palavras separadas.

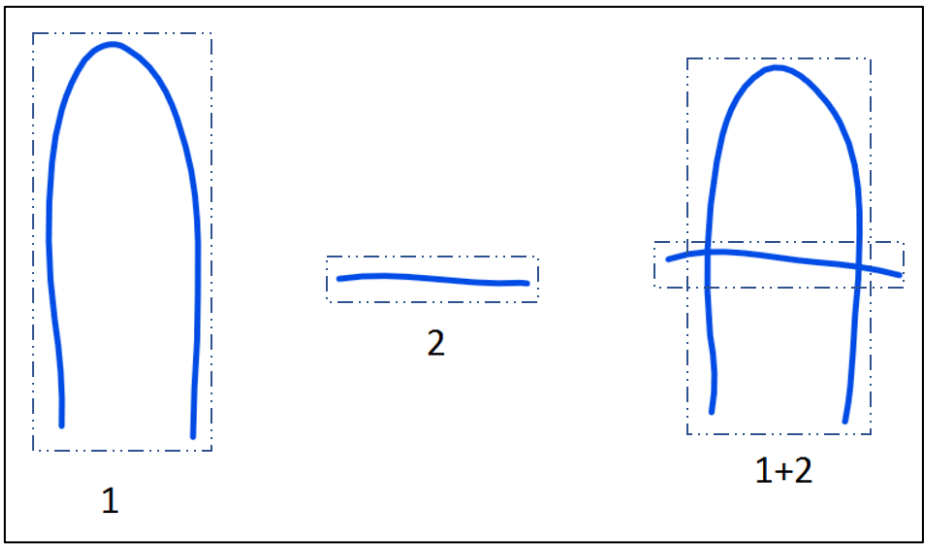

Figura 16 – Método para determinar a intersecção de strokes.

Uma vez que se era capaz de detectar espaços entre palavras, um novo problema a ser contornado surgiu: detectar quando um espaço entre letras tratava-se de um simples espaços entre letras de uma mesma palavra ou quando tratava-se de um espaçamento entre palavras. Para resolver tal problemática, fez-se um pequeno estudo estatístico acerca da distribuição de espaços durante a escrita natural. No exemplo abaixo, temos as palavras "Allan Holanda", mostrada na figura 17. Foi extraído, programaticamente, a distância entre cada palavra adjacente, sendo disponibilizadas na tabela 1, bem como algumas métricas estatísticas. O respectivo gráfico é mostrado na figura 18.

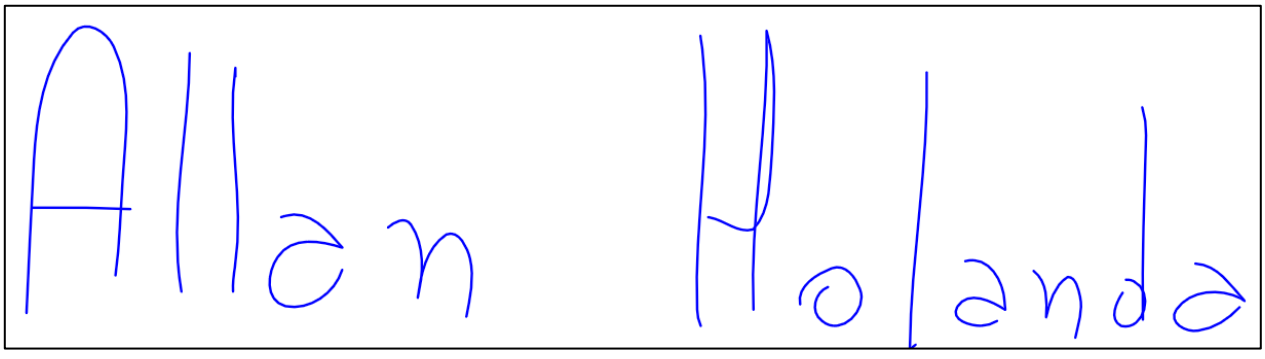

Figura 17 – Palavras "Allan Holanda" usadas no entendimento do modelo matemático a ser aplicado

Distância Original
112
43
28
92
239
10
71
81
40
60
62
24

Mínimo	10
Máximo	239
Média	72
Mediana	61
Desvio Padrão	58

Tabela 1 – Distâncias entre as letras da frase "Allan Holanda"

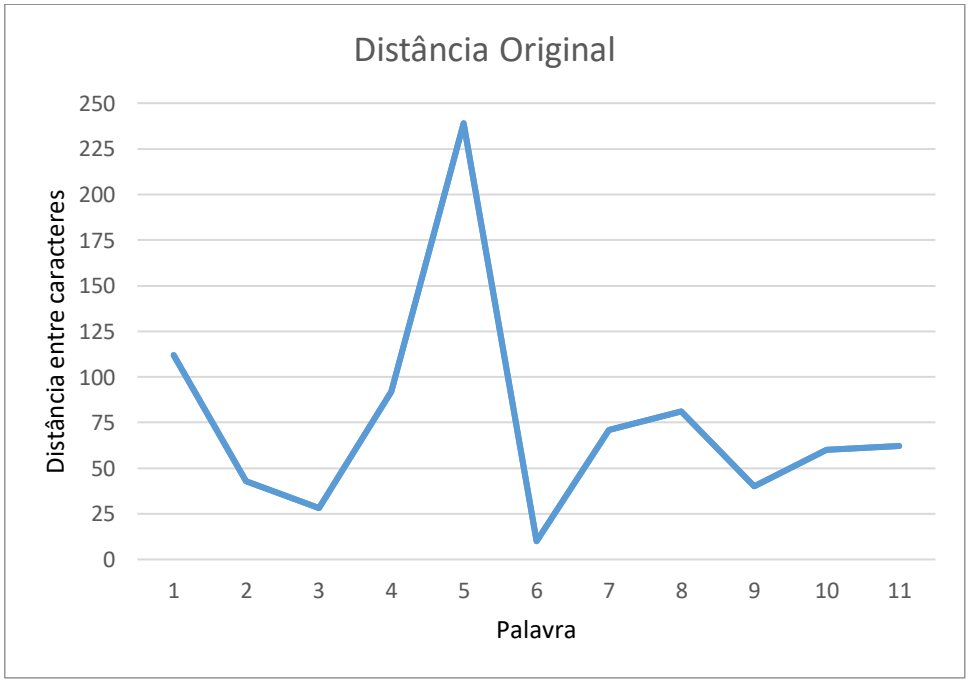

Figura 18 – Gráfico da distância entre palavras sem qualquer tratamento.

Percebe-se no gráfico acima que o pico entre espaços remete justamente ao espaçamento entre palavras. Para resolver o problema de diferenciação entre espaços, optou-se por usar como modelo matemático um filtro de mediana. Quando utiliza-se a mediana em uma janela de processamento de 5 itens, obteve-se os valores a seguir.

Distância Original	Distância Filtrada
112	112
43	92
28	92
92	43
239	71
10	81
71	71
81	60
40	62
60	60
62	40
24	24

Mínimo	10	24
Máximo	239	112
Média	72	67
Mediana	61	67
Desvio Padrão	58	24

Tabela 2 – Valores originais corrigidos por um filtro de Mediana

Nota-se que o valor de pico foi extremamente suavizado e normalizado de acordo com os seus vizinhos, mostrando-se um método bastante eficiente quando se tem como premissa que a quantidade de espaços pequenos é muito maior do que o espaço grande presente no espaçamento entre palavras. Um gráfico após a filtragem é mostrado na figura 19.

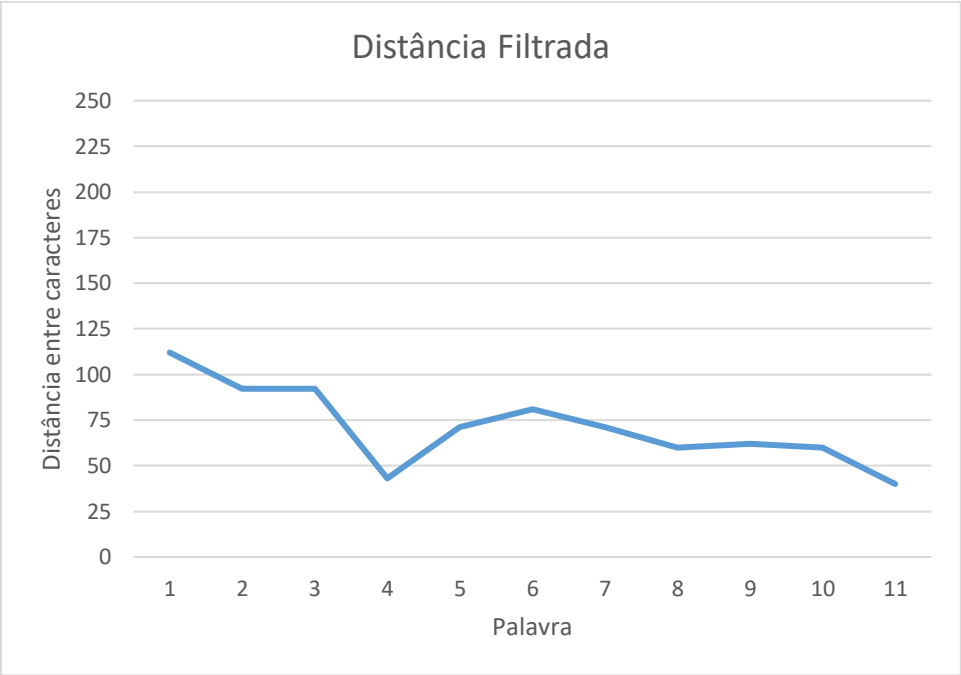

Figura 19 - Gráfico da distância entre palavras após a filtragem de mediana.

Logo após essa modelagem, determinou-se que o valor máximo da série filtrada seria o limiar entre um espaçamento entre letras de uma mesma palavra e o espaçamento entre letras de palavras diferentes. Após isso, pode-se posicionar as celas Braille logo abaixo do inicio de cada palavra, conforme demonstrado na figura 20 e 21.

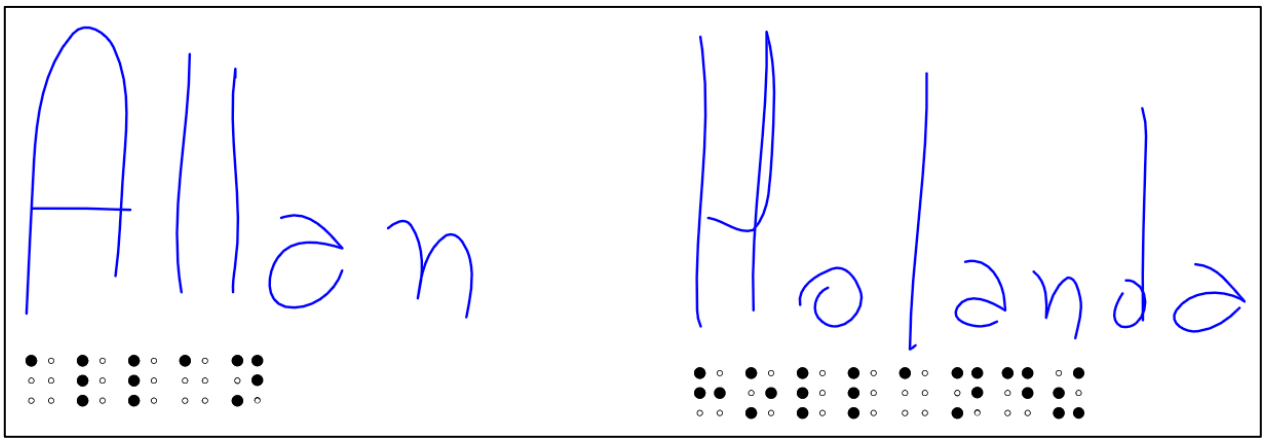

Figura 20 – Inserção de celas Braille logo após o início de cada palavra distinta.

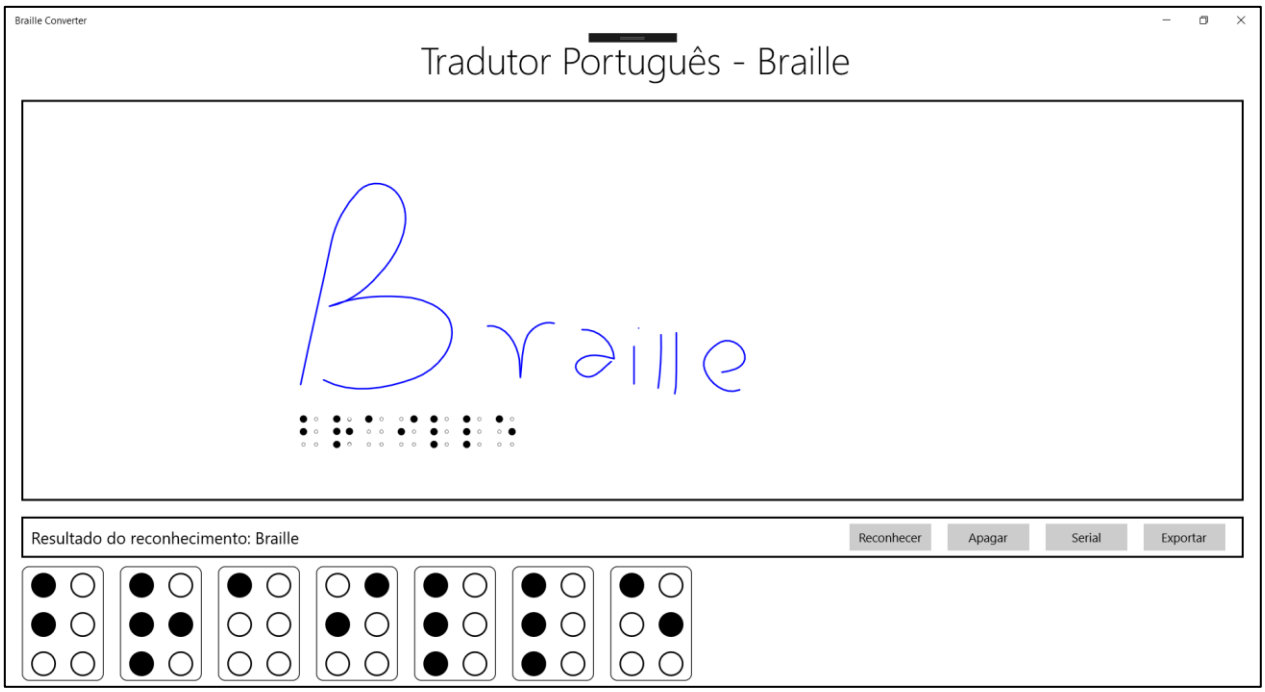

Figura 21 - Inserção de celas Braille logo após o início de cada palavra distinta vista no contexto da tela do programa.

Por fim, foi inserido no programa uma tela que suportasse a geração de documentos com toda uma sessão de escrita, capturando toda escrita natural na tela e convertendo-a em um histórico de palavras Braille, conforme mostra a figura 22.

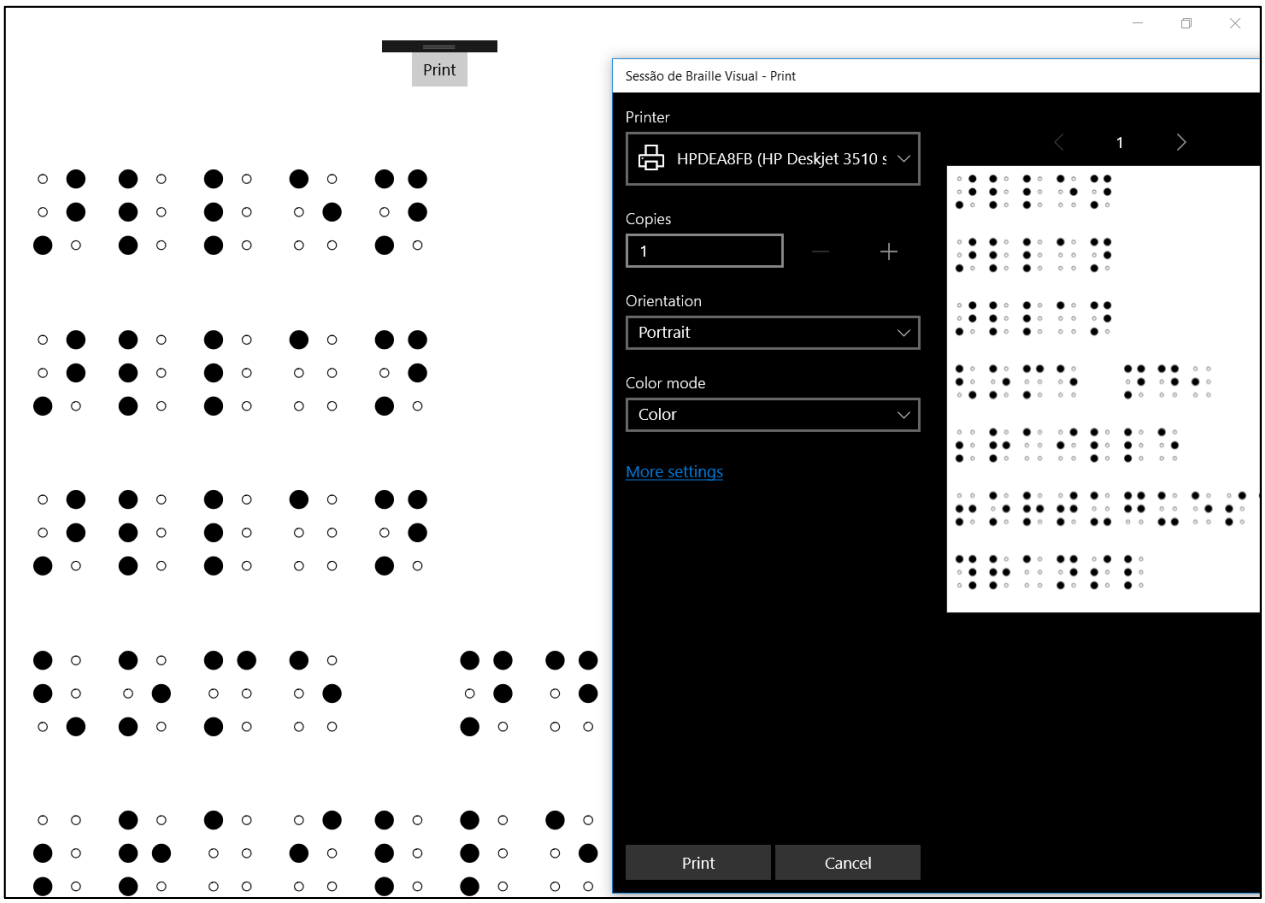

Figura 22 – Tela de geração de documentos e impressão de uma sessão de tradução Braille.

5. CRONOGRAMA DE EXECUÇÃO

O seguinte cronograma de execução foi planejado, contendo o que foi realizado e as atividades restantes do presente Trabalho de Graduação:

Atividades		2016							2017			
		Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr
1	Provas de conceito da implementação											
2	Produção do Relatório 1											
3	Busca de Literatura de Apoio											
4	Tradutor de Português para Braille Visual											
5	Reconhecimento de Escrita em Português											
6	Gerador de Documentos em Braille p/ Impressão											
7	Produção do Relatório 2											
8	Suporte a Extensibilidade e Futuras Expansões											
9	Testes Finais de usabilidade											
10	Produção do Relatório 3											

Legenda:

	Concluído
	Pendente/Planejado

6. REFERÊNCIAS

Apadev, Associação dos Pais e Amigos dos Deficientes Visuais. s.d. “O sistema Braile.” *apadev.* Acesso em 17 de Agosto de 2016. http://www.apadev.org.br/pages/workshop/Osistemabraile.pdf.

Holbrook, M., e L Nannen. 1997. “Methods of literacy instruction for Braille.” *Journal of Visual Impairment & Blindness* 420.

IBGE, Instituto Brasileiro de Geografia e Estatística. 2010. “Censo Demográfico.”

Microsoft Corporation. 2009. CONVERTING DIGITAL INK TO SHAPES AND TEXT. Washington, United States of America Patente US20100171754 A1. 09 de Abril.

Microsoft. 2016. *InkCanvas class.* Acesso em 16 de Agosto de 2016. https://msdn.microsoft.com/library/windows/apps/dn858535.

—. 2016. *Windows.UI.Input.Inking namespace.* Acesso em 16 de Agosto de 2016. https://msdn.microsoft.com/library/windows/apps/br208524.

Nicolaiewsky, Clarissa de Arruda, e Jane. Correa. 2008. “Escrita ortográfica e revisão de texto em Braille: uma história de reconstrução de paradigmas sobre o aprender.” *Scielo.* Acesso em 17 de Agosto de 2016. http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0101-32622008000200006&lng=en&nrm=iso.

Portal Brasil, Cidadania e Justiça. 2015. *Braile aumenta inclusão de cegos na sociedade.* 05 de Janeiro. Acesso em 16 de Agosto de 2016. http://www.brasil.gov.br/cidadania-e-justica/2015/01/braile-aumenta-inclusao-de-cegos-na-sociedade.

Pring, L. 1994. “Touch and go: learning to read Braille.” *Reading Research Quarterly, v. 29* 67-74.

Rao, Elsie. s.d. “Suggestions For Working With Braille Students In A Regular Classroom.” *Texas School for the Blind and Visually Impaired.* Acesso em 21 de Agosto de 2016.

http://www.tsbvi.edu/program-and-administrative-resources/3250-suggestions-for-working-with-braille-students-in-a-regular-classroom.

Techopedia. s.d. *Techopedia.* Acesso em 5 de Fevereiro de 2017. https://www.techopedia.com/definition/31623/optical-character-recognition-ocr.

The American Foundation for the Blind's Josephine L. Taylor Leadership Institute, Education Work Group. 2000. "Educating Students With Visual Impairments for Inclusion in Society: A Paper On The Inclusion Of Students With Visual Impairments." *AFB - American Foundation for the Blind.* 26 de Setembro. Acesso em 25 de Agosto de 2016. http://www.afb.org/info/programs-and-services/professional-development/teachers/inclusive-education/1235.

Whitney, Tyler. 2016. *Guide to Universal Windows Platform (UWP) apps.* 3 de Agosto. Acesso em 16 de Agosto de 2016. https://msdn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide.

7. ANEXOS

7.1. CÓDIGOS-FONTE

7.1.1. MainPage.xaml

```
<Page
    x:Class="TCC_Eng_Info.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:TCC_Eng_Info"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d">

    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
            <RowDefinition Height="50"/>
            <RowDefinition Height="160"/>
        </Grid.RowDefinitions>
        <TextBlock Grid.Row="0" x:Name="Header"
                   Text="Tradutor Português - Braille"
                   Style="{ThemeResource HeaderTextBlockStyle}"
                   Margin="10,0,0,0"
                       HorizontalAlignment="Center"/>
        <Border Grid.Row="1" BorderBrush="#FF000000" BorderThickness="2" Margin="20">
            <Grid Name="inkGrid">
                <InkCanvas Name="inkCanvas" />
            </Grid>
        </Border>
        <Grid  Grid.Row="2" BorderBrush="#FF000000" BorderThickness="2"
Margin="20,0,20,0">
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*"/>
                <ColumnDefinition Width="120"/>
                <ColumnDefinition Width="120"/>
                <ColumnDefinition Width="120"/>
                <ColumnDefinition Width="120"/>
            </Grid.ColumnDefinitions>
            <TextBlock Grid.Column="0" Name="RecognitionResult" TextAlignment="Left"
VerticalAlignment="Center" Margin="10,0,0,0" FontSize="20" />
            <Button Grid.Column="1" x:Name="RecognizeButton" Content="Reconhecer"
Width="100"/>
            <Button Grid.Column="2" x:Name="ClearButton" Content="Apagar" Width="100"/>
            <ToggleButton Grid.Column="3" x:Name="EnableSerialButton" Content="Serial"
Width="100"/>
            <Button Grid.Column="4" x:Name="ExportPDFButton" Content="Exportar"
Width="100"/>
        </Grid>
        <StackPanel Grid.Row="3" Name="BraillePanel"  HorizontalAlignment="Left"
Orientation="Horizontal" Margin="10,0,10,0" />
    </Grid>
</Page>
```

7.1.2. MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using Windows.UI;
using Windows.UI.Core;
using Windows.UI.Input.Inking;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using TCC_Eng_Info.Models;
using System.Text;
using Windows.Foundation;
using System.Linq;
using Windows.UI.Xaml.Media;

namespace TCC_Eng_Info
{
    /// <summary>
    /// Página principal do aplicativo tradutor.
    /// </summary>
    public sealed partial class MainPage : Page
    {
        private DispatcherTimer _recogTimer = null;
        private InkRecognizerContainer _inkRecognizerContainer = null;
        private SerialOutput _serialDevice = null;
        private StringBuilder _strBuilder = null;


        /// <summary>
        /// Construtor da página principal do aplicativo tradutor.
        /// </summary>
        public MainPage()
        {
            this.InitializeComponent();

            // Escolhe-se quais dispositivos deseja-se permitir como entrada.
            inkCanvas.InkPresenter.InputDeviceTypes =
                CoreInputDeviceTypes.Mouse |
                CoreInputDeviceTypes.Touch;

            // Inicializa-se os atributos de desenho do componente usado.
            var attributes = new InkDrawingAttributes()
            {
                Color = Colors.Blue,
                FitToCurve = true,
                IgnorePressure = true
            };
            inkCanvas.InkPresenter.UpdateDefaultDrawingAttributes(attributes);

            // Inicializa-se os eventos de clique.
            //RecognizeButton.Click += RecognizeButton_Click;
            ClearButton.Click += ClearButton_ClickAsync;
            EnableSerialButton.Click += EnableSerialButton_Click;
            ExportPDFButton.Click += ExportPDFButton_Click;

            // Evento disparado quando o usuário para de escrever na tela.
```

```
            inkCanvas.InkPresenter.StrokesCollected += inkCanvas_StrokesCollected;
            // Evento disparado quando o primeiro stroke é detectado.
            inkCanvas.InkPresenter.StrokeInput.StrokeStarted += inkCanvas_StrokeStarted;

            // Cria o container usado no reconhecimento manuscrito.
            _inkRecognizerContainer = new InkRecognizerContainer();

            // Timer de controle que gerencia o reconhecimento dinâmico.
            _recogTimer = new DispatcherTimer()
            {
                Interval = new TimeSpan(0, 0, 1)
            };
            _recogTimer.Tick += recoTimer_Tick;

            // Estrutura para guardar o texto da sessão de captura da linguagem natural.
            _strBuilder = new StringBuilder();
        }

        /// <summary>
        /// Navega até a página de exportação de uma sessão de texto Braille.
        /// </summary>
        /// <param name="sender">Objeto que disparou a tarefa.</param>
        /// <param name="e">Encapsulamento de parâmetros do evento.</param>
        private void ExportPDFButton_Click(object sender, RoutedEventArgs e)
        {
            Frame.Navigate(typeof(ExportPage), _strBuilder);
        }

        /// <summary>
        /// Botão que limpa os strokes do Canvas, as células Braille e o texto presente no
dispositivo conectado pela Serial.
        /// </summary>
        /// <param name="sender">Objeto que disparou a tarefa.</param>
        /// <param name="e">Encapsulamento de parâmetros do evento.</param>
        private async void ClearButton_ClickAsync(object sender, RoutedEventArgs e)
        {
            inkCanvas.InkPresenter.StrokeContainer.Clear();
            BraillePanel.Children.Clear();
            var tbs = inkGrid.Children.OfType<TextBlock>().ToList();
            foreach (var tb in tbs)
            {
                inkGrid.Children.Remove(tb);
            }
            RecognitionResult.Text = string.Empty;

            if (_serialDevice != null)
                await _serialDevice.SendTextToDevice('\n'.ToString());
        }

        /// <summary>
        /// Se o respectivo botão de toggle possuir o valor verdadeiro, os dados do
reconhecimento são enviados pela porta Serial.
        /// </summary>
        /// <param name="sender">Objeto que disparou a tarefa.</param>
        /// <param name="e">Encapsulamento de parâmetros do evento.</param>
        private void EnableSerialButton_Click(object sender, RoutedEventArgs e)
        {
```

```
            if (EnableSerialButton.IsChecked == true)
            {
                _serialDevice = new SerialOutput("COM4");
            }
            else
            {
                _serialDevice = null;
            }
        }

        /// <summary>
        /// Handler para o evento de tick do timer de controle.
        /// </summary>
        /// <param name="sender">Objeto que disparou a tarefa.</param>
        /// <param name="e">Encapsulamento de parâmetros do evento.</param>
        private void recoTimer_Tick(object sender, object e)
        {
            Recognize_Tick();
        }

        /// <summary>
        /// O evento é disparado quando o usuário inicia a escrita (usando o dedo ou o
mouse na tela).
        /// O timer de controle é parado para permitir que o usuário escreva de maneira
contínua.
        /// </summary>
        /// <param name="sender">Objeto que disparou a tarefa.</param>
        /// <param name="args">Encapsulamento de parâmetros do evento.</param>
        private void inkCanvas_StrokeStarted(InkStrokeInput sender, PointerEventArgs args)
        {
            _recogTimer.Stop();
        }

        /// <summary>
        /// O evento é disparado sempre que o usuário para de escrever (soltando o dedo ou
o mouse da tela).
        /// Após 1 segundo ou caso não haja nenhuma entrada do usuário, o reconhecimento
será iniciado.
        /// </summary>
        /// <param name="sender">Objeto que disparou a tarefa.</param>
        /// <param name="args">Encapsulamento de parâmetros do evento.</param>
        private void inkCanvas_StrokesCollected(InkPresenter sender,
InkStrokesCollectedEventArgs args)
        {
            _recogTimer.Start();
        }

        /// <summary>
        /// Quando o usuário para de escrever após 1 segundos, o reconhecimento se inicia.
        /// </summary>
        private async void Recognize_Tick()
        {
            // Pega todos os strokes do InkCanvas:
            IReadOnlyList<InkStroke> currentStrokes =
inkCanvas.InkPresenter.StrokeContainer.GetStrokes();

            // Garante que ao menos um stroke está presente:
```

```
            if (currentStrokes.Count > 0)
            {
                var positions = GetWordsPositions(currentStrokes, 30);

                if (!(_inkRecognizerContainer == null))
                {
                    // Reconhece todos os strokes presentes no canvas.
                    IReadOnlyList<InkRecognitionResult> recognitionResults =
                        await _inkRecognizerContainer.RecognizeAsync(
                            inkCanvas.InkPresenter.StrokeContainer,
                            InkRecognitionTarget.All);

                    if (recognitionResults.Count > 0)
                    {
                        int i = 0;
                        string text = string.Empty;
                        foreach (var result in recognitionResults)
                        {
                            // Pega-se todos os candidatos de cada resultado reconhecido
possível:
                            IReadOnlyList<string> candidates = result.GetTextCandidates();
                            // Admite-se o primeiro de cada um como sendo o correto:
                            text += candidates[0] + " ";

                            TextBlock tb = new TextBlock()
                            {
                                //Utiliza-se a fonte Braille AOE presente na pasta de
Assets para renderizar o PDF.
                                FontFamily = new FontFamily("../Assets/Fonts/Braille AOE
Font.TTF#Braille AOE"),
                                FontSize = 50,
                                Name = $"Block{i}",
                                Text = candidates[0].ToString().ToLower()
                            };

                            //Adiciona as celas Braille abaixo de cada palavra.
                            inkGrid.Children.Add(tb);
                            PlaceText(tb, positions[i]);
                            i++;
                        }
                        // Mostra o resultado do reconhecimento:
                        RecognitionResult.Text = "Resultado do reconhecimento: " + text;
                        //Envia conteúdo do texto via Serial:
                        if (_serialDevice != null)
                            await _serialDevice.SendTextToDevice(text + '\n');

                        //Adiciona as celas Braille a tela
                        AddBrailleCells(text);

                        _strBuilder.AppendLine(text);

                        // Limpa o Canvas uma vez que o reconhecimento foi finalizado:
                        //inkCanvas.InkPresenter.StrokeContainer.Clear();
                    }
                    else
                    {
```

```
                        RecognitionResult.Text = "Nenhum resultado proveniente do
reconhecimento!";
                    }
                }
                else
                {
                    Windows.UI.Popups.MessageDialog messageDialog = new
Windows.UI.Popups.MessageDialog("Instale alguma engine de reconhecimento!");
                    await messageDialog.ShowAsync();
                }
            }
            else
            {
                RecognitionResult.Text = "Nenhum resultado proveniente do
reconhecimento!";
            }

            // Para o timer e aguarda algum stroke ser desenhado na tela novamente.
            _recogTimer.Stop();
        }

        /// <summary>
        /// Posiciona o texto em baixo do conjunto correto de strokes
        /// </summary>
        private void PlaceText(TextBlock tb, Thickness position)
        {
            tb.Margin = position;
        }

        /// <summary>
        /// Obtem as posições que as palavras Braille serão posicionadas no Ink.
        /// </summary>
        /// <param name="strokes">Conjunto de strokes pertencentes a de um Ink usado em
reconhecimento de escrita natural.</param>
        /// <param name="marginTop">Margem superior a ser adicionada à posição.</param>
        /// <returns></returns>
        private List<Thickness> GetWordsPositions(IReadOnlyList<InkStroke> strokes, double
marginTop)
        {
            var positions = new List<Thickness>
            {
                getThickness(strokes[0], marginTop)
            };

            var filter = new MedianFilter(3);
            var diffs = GetDiffs(strokes);
            var widthMed = filter.Filter(diffs).Max();

            for (int i = 1; i < strokes.Count; i++)
            {
                var rect1 = strokes[i - 1].BoundingRect;
                var rect2 = strokes[i].BoundingRect;
                rect1.Intersect(rect2);
                if (rect1.IsEmpty)
                {
                    var diff = strokes[i].BoundingRect.Left - strokes[i -
1].BoundingRect.Left;
```

```
                    if (diff > widthMed)
                    {
                        positions.Add(getThickness(strokes[i], marginTop));
                    }
                }
            }
            return positions;
        }

        /// <summary>
        /// Função utilizada para pegar a diferença entre letras adjacentes
        /// </summary>
        /// <param name="strokes">Conjunto de strokes pertencentes a de um Ink usado em
reconhecimento de escrita natural.</param>
        /// <returns></returns>
        private List<double> GetDiffs(IReadOnlyList<InkStroke> strokes)
        {
            var diffs = new List<double>();

            for (int i = 1; i < strokes.Count; i++)
            {
                var rect1 = strokes[i - 1].BoundingRect;
                var rect2 = strokes[i].BoundingRect;
                rect1.Intersect(rect2);
                if (rect1.IsEmpty)
                {
                    diffs.Add(strokes[i].BoundingRect.Left - strokes[i -
1].BoundingRect.Left);
                }
            }

            return diffs;
        }

        /// <summary>
        /// Função utilizada somente no estudo estatistico de distância entre palavras.
        /// </summary>
        /// <param name="strokes">Conjunto de strokes pertencentes a de um Ink usado em
reconhecimento de escrita natural.</param>
        /// <returns></returns>
        private string GetWordsSpaces(IReadOnlyList<InkStroke> strokes)
        {
            var positions = "";

            for (int i = 1; i < strokes.Count; i++)
            {
                var rect1 = strokes[i - 1].BoundingRect;
                var rect2 = strokes[i].BoundingRect;
                rect1.Intersect(rect2);
                if (rect1.IsEmpty)
                {
                    var diff = strokes[i].BoundingRect.Left - strokes[i -
1].BoundingRect.Left;
                    positions += diff + ",";
                }
            }
            return positions;
```

```
        }

        /// <summary>
        /// Extrai a posição mais abaixo e a esquerda de um stroke.
        /// </summary>
        /// <param name="stroke">Risco a ser analisado.</param>
        /// <param name="marginTop">Margem superior a ser adicionada à posição.</param>
        /// <returns></returns>
        private Thickness getThickness(InkStroke stroke, double marginTop)
        {
            var left = stroke.BoundingRect.Left;
            var top = stroke.BoundingRect.Top;
            var height = stroke.BoundingRect.Height;

            return new Thickness(left, top + height + marginTop, 0, 0);
        }

        /// <summary>
        /// Adiciona um conjunto de células Braille à View correspondentes a um texto em
português.
        /// </summary>
        /// <param name="text">Texto a ser convertido para Braille</param>
        private void AddBrailleCells(string text)
        {
            BraillePanel.Children.Clear();

            var letters = text.ToUpper().ToCharArray();

            foreach (var letter in letters)
            {
                var cell = new BrailleCell(letter);
                BraillePanel.Children.Add(cell);
            }
        }
    }
}
```

7.1.3. BraillePoint.cs

```
using Windows.UI.Xaml.Media;

namespace TCC_Eng_Info
{
    public class BraillePoint
    {
        public SolidColorBrush Color { get; set; }
    }
}
```

7.1.4. BrailleCell.xaml

```
<UserControl
    x:Class="TCC_Eng_Info.BrailleCell"
```

```
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:local="using:TCC_Eng_Info"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="160"
    d:DesignWidth="120">
    <Border x:Name="CellContainer"  BorderBrush="#FF000000" BorderThickness="1"
CornerRadius="8" Height="140" Width="100" Margin="10">
        <GridView x:Name="Cell" VerticalContentAlignment="Center"
HorizontalContentAlignment="Center" SelectedIndex="-1" SelectionMode="None"
ScrollViewer.HorizontalScrollBarVisibility="Hidden"
ScrollViewer.VerticalScrollBarVisibility="Hidden">
            <GridView.ItemTemplate>
                <DataTemplate>
                    <Ellipse
                        Fill="{Binding Color}"
                        Height= "30"
                        Width="30"
                        Margin="8,0,0,0"
                        StrokeThickness="2"
                        Stroke="Black"/>
                </DataTemplate>
            </GridView.ItemTemplate>
        </GridView>
    </Border>
</UserControl>
```

7.1.5. BrailleCell.xaml.cs

```
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;

namespace TCC_Eng_Info
{
    public sealed partial class BrailleCell : UserControl
    {
        public BrailleCell(char letter)
        {
            this.InitializeComponent();

            if (letter != ' ')
                Cell.ItemsSource = BrailleUtil.GetBrailleCell(letter);
            else
                CellContainer.BorderThickness = new Thickness(0);
        }

    }
}
```

7.1.6. BrailleUtil.cs

```
using System.Collections.Generic;
using Windows.UI;
using Windows.UI.Xaml.Media;

namespace TCC_Eng_Info
{
    public static class BrailleUtil
    {
        public static List<BraillePoint> GetBrailleCell(char letter)
        {
            var brailleCell = new List<BraillePoint>(6);

            var braillePoints = LetterBrailleConverter.LetterToBraille(letter);

            foreach (var point in braillePoints)
                brailleCell.Add(GetPointColors(point));

            return brailleCell;
        }

        private static BraillePoint GetPointColors(bool point)
        {
            if (point)
                return new BraillePoint() { Color = new
SolidColorBrush(Color.FromArgb(255, 0, 0, 0)) };
            else
                return new BraillePoint() { Color = new
SolidColorBrush(Color.FromArgb(255, 255, 255, 255)) };
        }

    }
}
```

7.1.7. LetterBrailleConverter.cs

```
namespace TCC_Eng_Info
{
    public static class LetterBrailleConverter
    {
        private static readonly bool[] A = new bool[] { true, false, false, false, false,
false };
        private static readonly bool[] B = new bool[] { true, false, true, false, false,
false };
        private static readonly bool[] C = new bool[] { true, true, false, false, false,
false };
        private static readonly bool[] D = new bool[] { true, true, false, true, false,
false };
        private static readonly bool[] E = new bool[] { true, false, false, true, false,
false };
        private static readonly bool[] F = new bool[] { true, true, true, false, false,
false };
```

```
        private static readonly bool[] G = new bool[] { true, true, true, true, false,
false };
        private static readonly bool[] H = new bool[] { true, false, true, true, false,
false };
        private static readonly bool[] I = new bool[] { false, true, true, false, false,
false };
        private static readonly bool[] J = new bool[] { false, true, true, true, false,
false };
        private static readonly bool[] K = new bool[] { true, false, false, false, true,
false };
        private static readonly bool[] L = new bool[] { true, false, true, false, true,
false };
        private static readonly bool[] M = new bool[] { true, true, false, false, true,
false };
        private static readonly bool[] N = new bool[] { true, true, false, true, true,
false };
        private static readonly bool[] O = new bool[] { true, false, false, true, true,
false };
        private static readonly bool[] P = new bool[] { true, true, true, false, true,
false };
        private static readonly bool[] Q = new bool[] { true, true, true, true, true,
false };
        private static readonly bool[] R = new bool[] { true, false, true, true, true,
false };
        private static readonly bool[] S = new bool[] { false, true, true, false, true,
false };
        private static readonly bool[] T = new bool[] { false, true, true, true, true,
false };
        private static readonly bool[] U = new bool[] { true, false, false, false, true,
true };
        private static readonly bool[] V = new bool[] { true, false, true, false, true,
true };
        private static readonly bool[] W = new bool[] { false, true, true, true, false,
true };
        private static readonly bool[] X = new bool[] { true, true, false, false, true,
true };
        private static readonly bool[] Y = new bool[] { true, true, false, true, true,
true };
        private static readonly bool[] Z = new bool[] { true, false, false, true, true,
true };

        private static readonly bool[] Dot = new bool[] { false, false, true, true, false,
true };
        private static readonly bool[] Comma = new bool[] { false, false, true, false,
false, false };
        private static readonly bool[] Semicolon = new bool[] { false, false, true, false,
true, false };
        private static readonly bool[] Hyphen = new bool[] { false, false, false, false,
true, true };
        private static readonly bool[] Question = new bool[] { false, false, true, false,
false, true };
        private static readonly bool[] Exclamation = new bool[] { false, false, true,
true, true, false };

        private static readonly bool[] Empty = new bool[] { false, false, false, false,
false, false };
```

```
public static bool[] LetterToBraille(char letter)
{
    switch (letter)
    {
        case 'A':
            return A;
        case 'B':
            return B;
        case 'C':
            return C;
        case 'D':
            return D;
        case 'E':
            return E;
        case 'F':
            return F;
        case 'G':
            return G;
        case 'H':
            return H;
        case 'I':
            return I;
        case 'J':
            return J;
        case 'K':
            return K;
        case 'L':
            return L;
        case 'M':
            return M;
        case 'N':
            return N;
        case 'O':
            return O;
        case 'P':
            return P;
        case 'Q':
            return Q;
        case 'R':
            return R;
        case 'S':
            return S;
        case 'T':
            return T;
        case 'U':
            return U;
        case 'V':
            return V;
        case 'W':
            return W;
        case 'X':
            return X;
        case 'Y':
            return Y;
        case 'Z':
            return Z;
```

```
                case '.':
                    return Dot;
                case ',':
                    return Comma;
                case ';':
                    return Semicolon;
                case '-':
                    return Hyphen;
                case '?':
                    return Question;
                case '!':
                    return Exclamation;

                default:
                    return Empty;
            }
        }
    }
}
```

7.1.1. MedianFilter.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TCC_Eng_Info.Models
{
    ///<Summary>
    /// Ferramenta que aplica o filtro de Mediana a uma lista.
    ///</Summary>
    public class MedianFilter
    {
        int _window;

        /// <summary>
        /// Construtor da ferramenta com o valor padrão de window = 3.
        /// </summary>
        public MedianFilter():this(3)
        {
        }

        /// <summary>
        /// Construtor da ferramenta.
        /// </summary>
        /// <param name="window">Tamanho da janela de quantidade de itens a serem
analisados ao mesmo tempo pela mediana.</param>
        public MedianFilter(int window)
        {
            _window = window;
        }
```

```
        /// <summary>
        /// Filtro que realiza uma suavização na curva dos dados.
        /// </summary>
        /// <param name="input">Lista de valores double a serem suavizados pelo
filtro.</param>
        /// <returns></returns>
        public List<double> Filter(List<double> input)
        {
            var output = new List<double>();
            var newInput = new List<double>(input.ToArray());

            for (int i = 0; i < _window - 2; i++)
            {
                newInput.Insert(0, input[0]);
                newInput.Insert(newInput.Count - 1, input[input.Count - 1]);
            }

            for (int i = 0; i < input.Count; i++)
            {
                var elem = Median(newInput.Skip(i).Take(_window).ToList());
                output.Add(elem);
            }

            return output;
        }

        /// <summary>
        /// Método que aplica a mediana sobre um conjunto de dados de uma janela.
        /// </summary>
        /// <param name="input">Lista de dados de entrada do processamento</param>
        /// <returns></returns>
        private double Median(List<double> input)
        {
            if (input.Count == 1) return input[0];

            input.Sort();
            if (input.Count % 2 == 0)
            {
                return input.Sum() / input.Count;
            }
            else
            {
                var mid = input.Count / 2;
                return input[mid];
            }
        }
    }
}
```